Assignment - 2

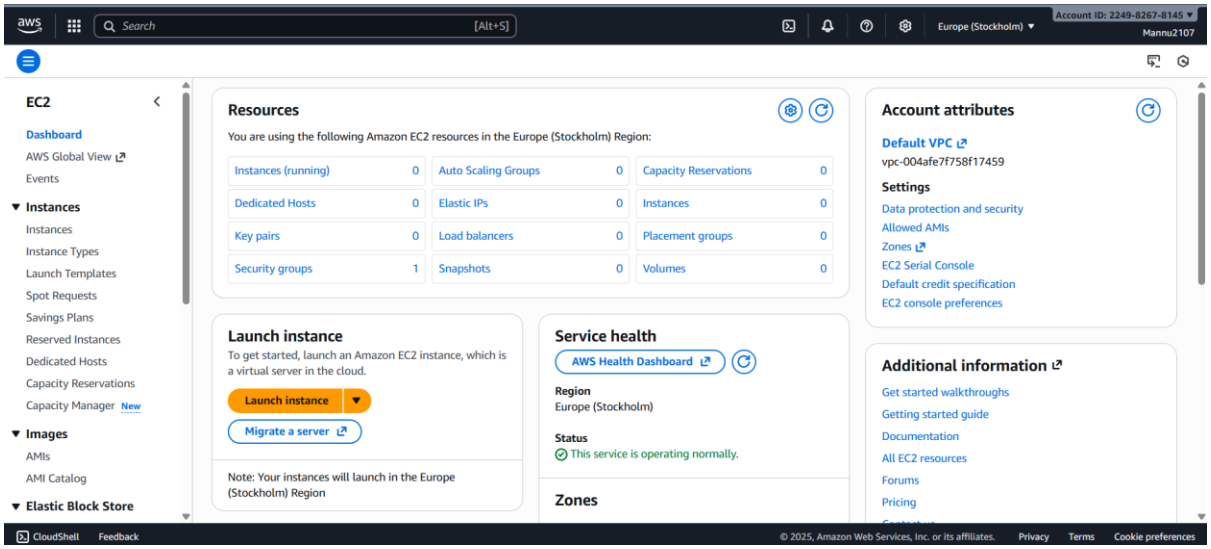
1. Using Docker Containerize sample application

a. By using AWS services

b. from the local machine

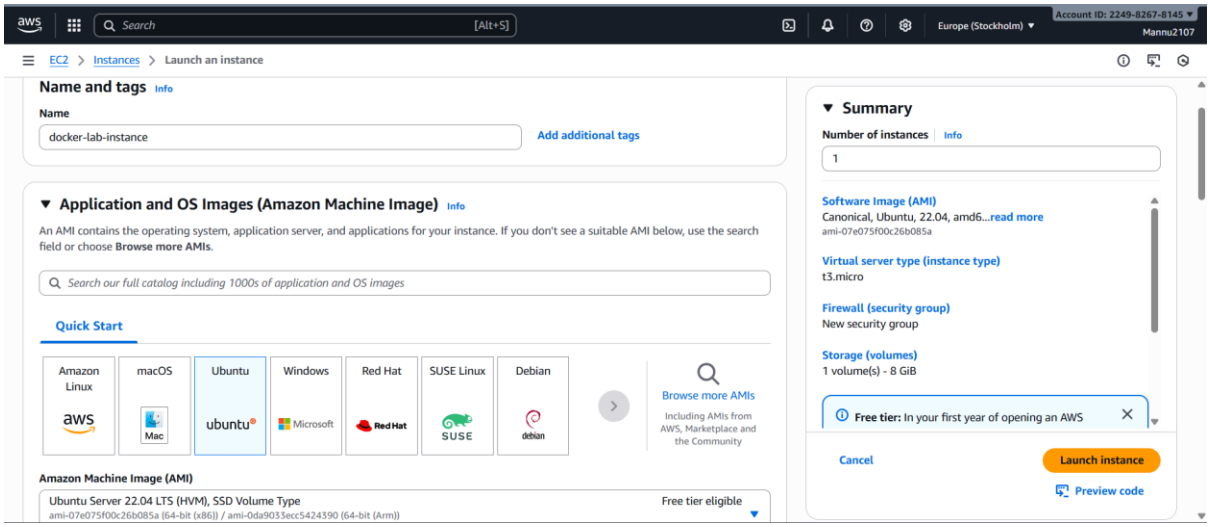
Using Docker Containerize sample application by using AWS services:-

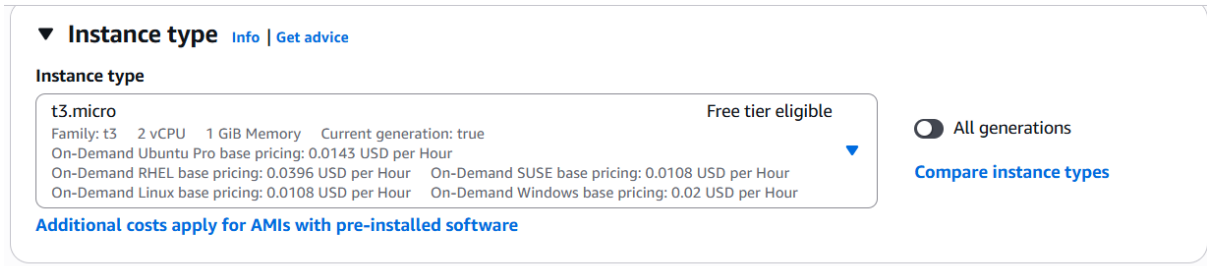
Step 1: Open the AWS Console → **EC2** → **Instances** → click **Launch instance**.

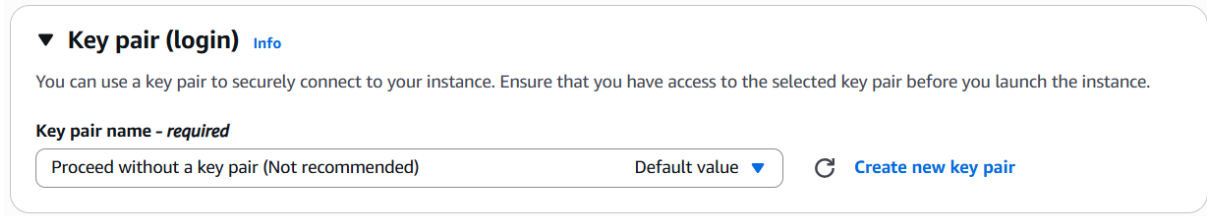


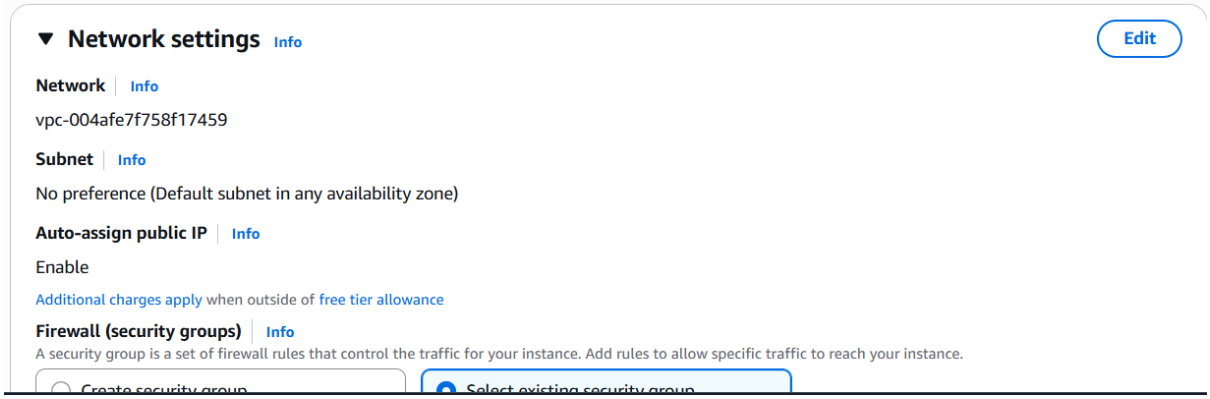
Step 2:

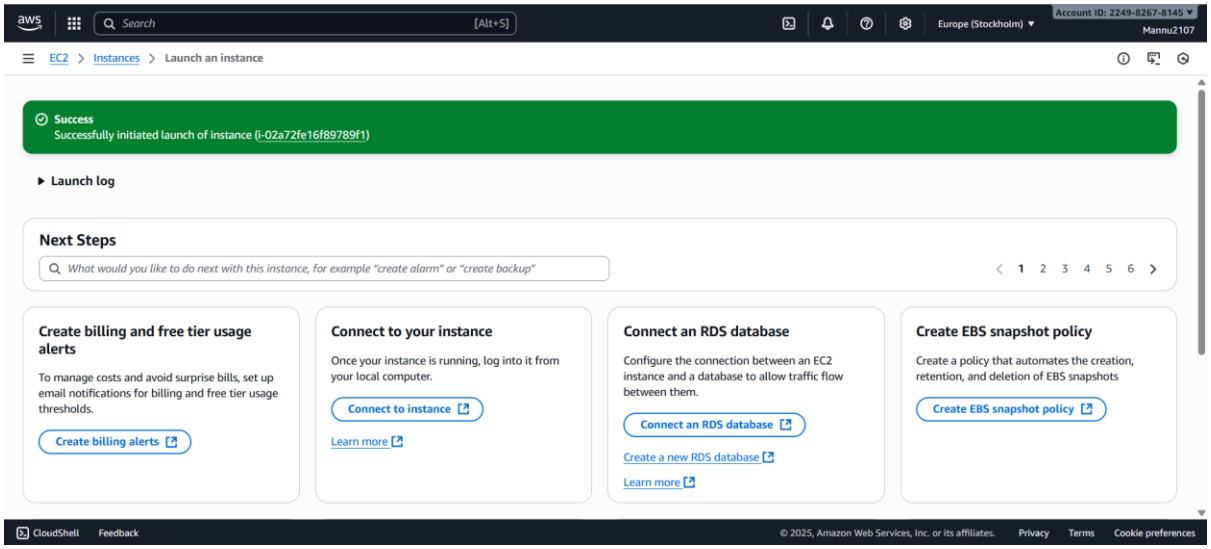
- Name: `docker-lab-instance`
- AMI: Select **Amazon Linux 2** or **Ubuntu 22.04**.
- Instance type: `t2.micro` (Free-tier eligible).
- Key pair: Create or use an existing one.
- Security group:
- Allow **SSH (22)** for connecting.
- Allow **HTTP (80)** for testing your app.
- Click **Launch instance**


Search
[Alt+S]
Europe (Stockholm)
Account ID: 2249-8267-8145
Mannu2107
EC2 > Instances > Launch an instance
Name and tags Info
Name
docker-lab-instance
Add additional tags
Application and OS Images (Amazon Machine Image) Info
An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose Browse more AMIs.
Search our full catalog including 1000s of application and OS images
Quick Start
Amazon Linux
macOS
Ubuntu
Windows
Red Hat
SUSE Linux
Debian
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community
Amazon Machine Image (AMI)
Ubuntu Server 22.04 LTS (HVM), SSD Volume Type
Free tier eligible
Summary
Number of instances Info
1
Software Image (AMI)
Canonical, Ubuntu, 22.04, amd6...read more
ami-07e075f00c26b085a
Virtual server type (instance type)
t3.micro
Firewall (security group)
New security group
Storage (volumes)
1 volume(s) - 8 GiB
Free tier: In your first year of opening an AWS
Cancel
Launch instance
Preview code

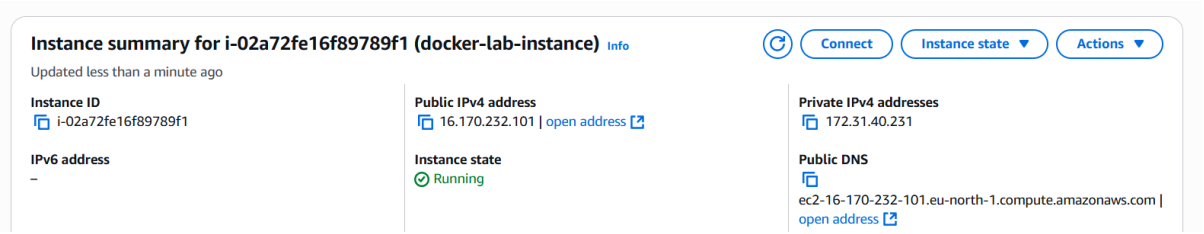

Instance type Info | Get advice
Instance type
t3.micro
Free tier eligible
Family: t3 2 vCPU 1 GiB Memory Current generation: true
On-Demand Ubuntu Pro base pricing: 0.0143 USD per Hour
On-Demand RHEL base pricing: 0.0396 USD per Hour
On-Demand SUSE base pricing: 0.0108 USD per Hour
On-Demand Linux base pricing: 0.0108 USD per Hour
On-Demand Windows base pricing: 0.02 USD per Hour
All generations
Compare instance types
Additional costs apply for AMIs with pre-installed software


Key pair (login) Info
You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.
Key pair name - required
Proceed without a key pair (Not recommended)
Default value
Create new key pair

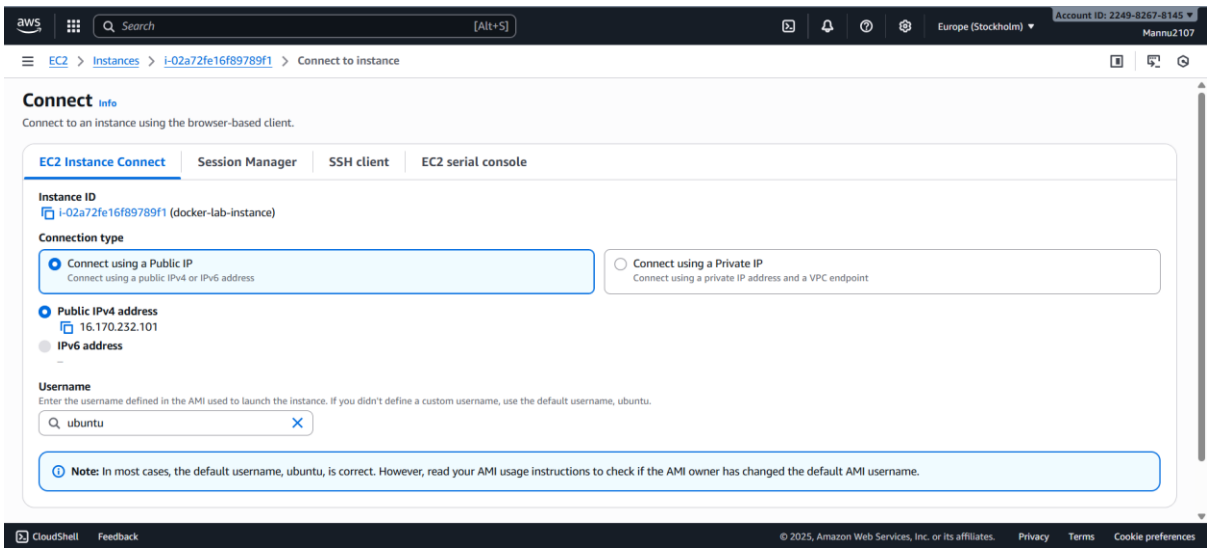

Network settings Info
Edit
Network Info
vpc-004afe7f758f17459
Subnet Info
No preference (Default subnet in any availability zone)
Auto-assign public IP Info
Enable
Additional charges apply when outside of free tier allowance
Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

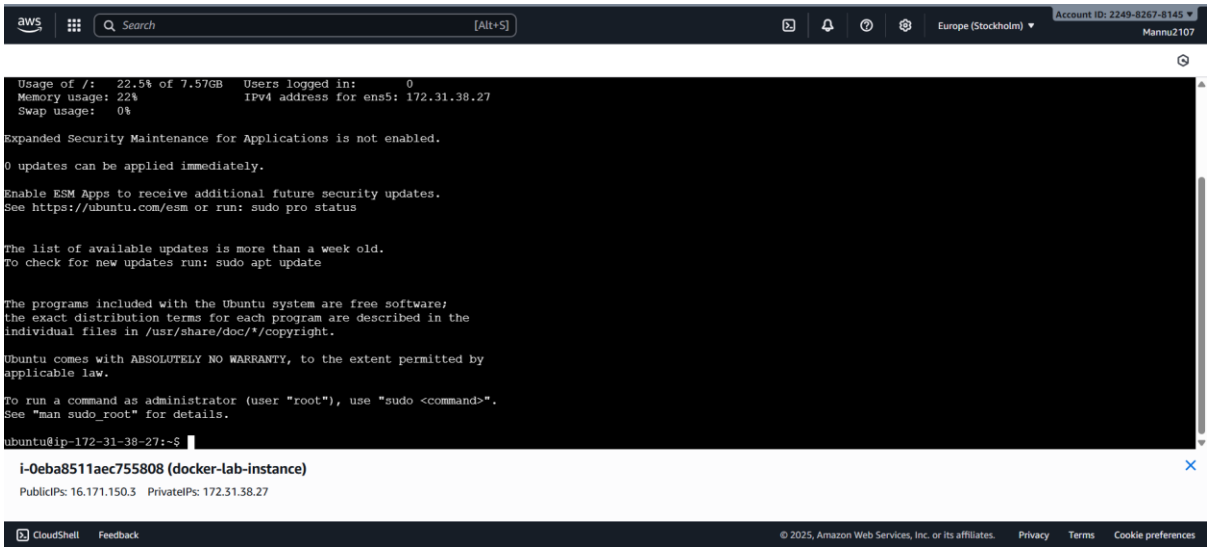


Step 3: After the instance is running, select it → click **Connect**.



Step 4: Choose **EC2 Instance Connect (browser-based)** → click **Connect**.





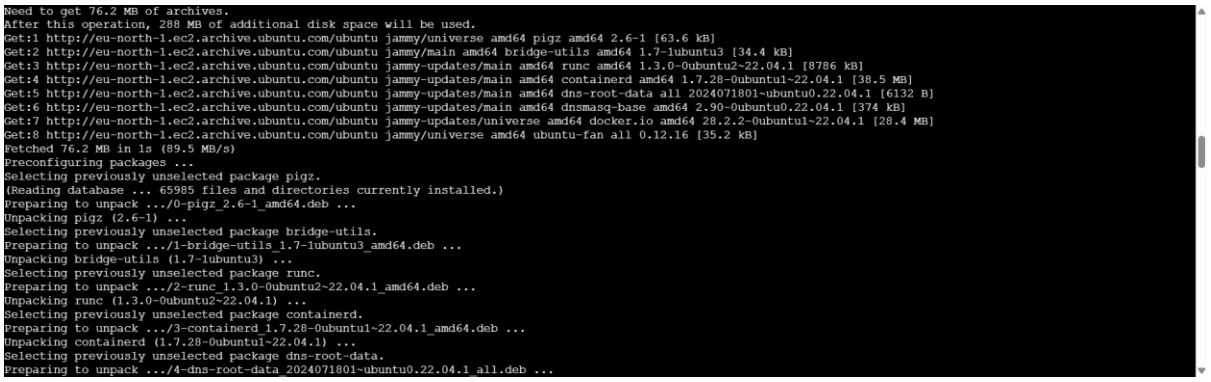
Step 5: Run the following commands in your EC2 terminal in order to update package manager, install docker, start docker service, and verify the docker installation.

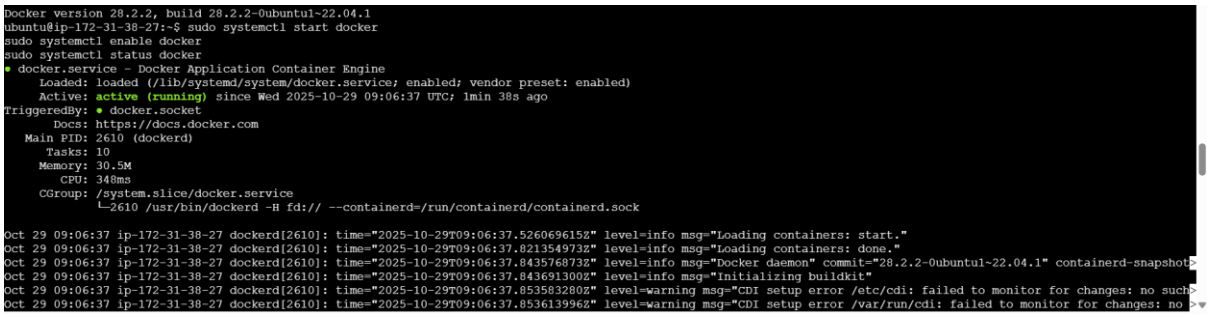
sudo apt update -y # (or sudo yum update -y for Amazon Linux)

sudo apt install docker.io -y # Ubuntu

sudo systemctl start docker sudo systemctl enable docker

docker –version





Step 5: Add Permissions. Then log out and reconnect for the permission to take effect.

sudo usermod -aG docker ubuntu # or ubuntu

```
ubuntu@ip-172-31-38-27:~$ sudo usermod -aG docker ubuntu
ubuntu@ip-172-31-38-27:~$
```

i-0eba8511aec755808 (docker-lab-instance)

PublicIPs: 16.171.150.3 PrivateIPs: 172.31.38.27

```
ubuntu@ip-172-31-38-27:~$ groups
ubuntu adm dialout cdrom floppy sudo audio dip video plugdev netdev lxd docker
ubuntu@ip-172-31-38-27:~$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
ubuntu@ip-172-31-38-27:~$
```

i-0eba8511aec755808 (docker-lab-instance)

PublicIPs: 16.171.150.3 PrivateIPs: 172.31.38.27

Step 6: You can create a simple Node.js or Python app.

mkdir docker-sample && cd docker-sample

nano app.js

```
  GNU nano 6.2                          app.js *
const http = require('http');
const port = 80;
const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello from Docker on AWS!\n');
});
server.listen(port, () => {
  console.log(`Server running at port ${port}`);
});

File Name to Write: app.js
^G Help          M-D DOS Format     M-A Append         M-B Backup File
^C Cancel        M-M Mac Format     M-P Prepend        ^T Browse
```

Step 7: Install Node.js

sudo apt install nodejs npm -y

```
Scanning processes...
Scanning candidates...
Scanning linux images...

Running kernel seems to be up-to-date.

Restarting services...
 /etc/needrestart/restart.d/systemd-manager
 systemctl restart acpid.service chrony.service containerd.service cron.service i
rqbalance.service multipathd.service packagekit.service polkit.service rsyslog.se
rvice serial-getty@ttyS0.service ssh.service systemd-journald.service systemd-net
workd.service systemd-resolved.service systemd-udevd.service
Service restarts being deferred:
 /etc/needrestart/restart.d/dbus.service
 systemctl restart docker.service
 systemctl restart getty@tty1.service
 systemctl restart networkd-dispatcher.service
 systemctl restart systemd-logind.service
 systemctl restart unattended-upgrades.service
 systemctl restart user@1000.service

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-38-27:~/docker-sample$
```

i-0eba8511aec755808 (docker-lab-instance)

PublicIPs: 16.171.150.3 PrivateIPs: 172.31.38.27

Step 8: Test it locally. Then open your **EC2 public IPv4 address** in a browser — you should see “Hello from Docker on AWS!”

sudo node app.js

```
ubuntu@ip-172-31-38-27:~/docker-sample$ sudo node app.js
Server running at port 80
```

i-0eba8511aec755808 (docker-lab-instance)

PublicIPs: 16.171.150.3 PrivateIPs: 172.31.38.27

Not secure 16.171.150.3

Arcade Leaderboard Homepage - myCo... Attendance Portal -... All Bookmarks

```
Hello from Docker on AWS!
```

Step 9: Create a file called **Dockerfile** and add the following.

nano Dockerfile

```
  GNU nano 6.2                        Dockerfile *
FROM node:18
WORKDIR /app
COPY . .
EXPOSE 80
CMD ["node", "app.js"]

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

i-0eba8511aec755808 (docker-lab-instance)

PublicIPs: 16.171.150.3 PrivateIPs: 172.31.38.27

Step 10: Build the Docker Image and verify the image.

docker build -t docker-sample-app .

docker images

```
ubuntu@ip-172-31-38-27:~/docker-sample$ docker build -t docker-sample-app .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  3.072kB
Step 1/5 : FROM node:18
18: Pulling from library/node
3e6b9d1a9511: Pulling fs layer
37927ed901b1: Pulling fs layer
79b2f47ad444: Pulling fs layer
e23f099911d6: Pulling fs layer
cda7f44f2bdd: Pulling fs layer
c6b30c3f1696: Pulling fs layer
3697be50c98b: Pulling fs layer
461077a72fb7: Pulling fs layer
cda7f44f2bdd: Waiting
e23f099911d6: Waiting
c6b30c3f1696: Waiting
3697be50c98b: Waiting
461077a72fb7: Waiting
37927ed901b1: Verifying Checksum
37927ed901b1: Download complete
3e6b9d1a9511: Verifying Checksum
3e6b9d1a9511: Download complete
79b2f47ad444: Verifying Checksum
79b2f47ad444: Download complete
cda7f44f2bdd: Verifying Checksum
cda7f44f2bdd: Download complete
3697be50c98b: Verifying Checksum
3697be50c98b: Download complete
c6b30c3f1696: Verifying Checksum
c6b30c3f1696: Download complete
461077a72fb7: Verifying Checksum
461077a72fb7: Download complete
```

i-0eba8511aec755808 (docker-lab-instance)

PublicIPs: 16.171.150.3 PrivateIPs: 172.31.38.27

```
461077a72fb7: Verifying Checksum
461077a72fb7: Download complete
e23f099911d6: Verifying Checksum
e23f099911d6: Download complete
3e6b9d1a9511: Pull complete
37927ed901b1: Pull complete
79b2f47ad444: Pull complete
e23f099911d6: Pull complete
cda7f44f2bdd: Pull complete
c6b30c3f1696: Pull complete
3697be50c98b: Pull complete
461077a72fb7: Pull complete
Digest: sha256:c6ae79e38498325db67193d391e6ec1d224d96c693a8a4d943498556716d3783
Status: Downloaded newer image for node:18
 ---> b50082bc3670
Step 2/5 : WORKDIR /app
 ---> Running in 85ddf8854509
 ---> Removed intermediate container 85ddf8854509
 ---> 8c87297ac485
Step 3/5 : COPY . .
 ---> 645d061c1903
Step 4/5 : EXPOSE 80
 ---> Running in 69d5454bb8db
 ---> Removed intermediate container 69d5454bb8db
 ---> 8875d14934e4
Step 5/5 : CMD ["node", "app.js"]
 ---> Running in 4b76a71b203e
 ---> Removed intermediate container 4b76a71b203e
 ---> ac06fbda8386
Successfully built ac06fbda8386
Successfully tagged docker-sample-app:latest
ubuntu@ip-172-31-38-27:~/docker-sample$ docker images
REPOSITORY          TAG       IMAGE ID       CREATED          SIZE
docker-sample-app   latest    ac06fbda8386   10 seconds ago   1.09GB
node                18        b50082bc3670   7 months ago     1.09GB
ubuntu@ip-172-31-38-27:~/docker-sample$
```

Step 11: Run the Docker Container and check running containers.

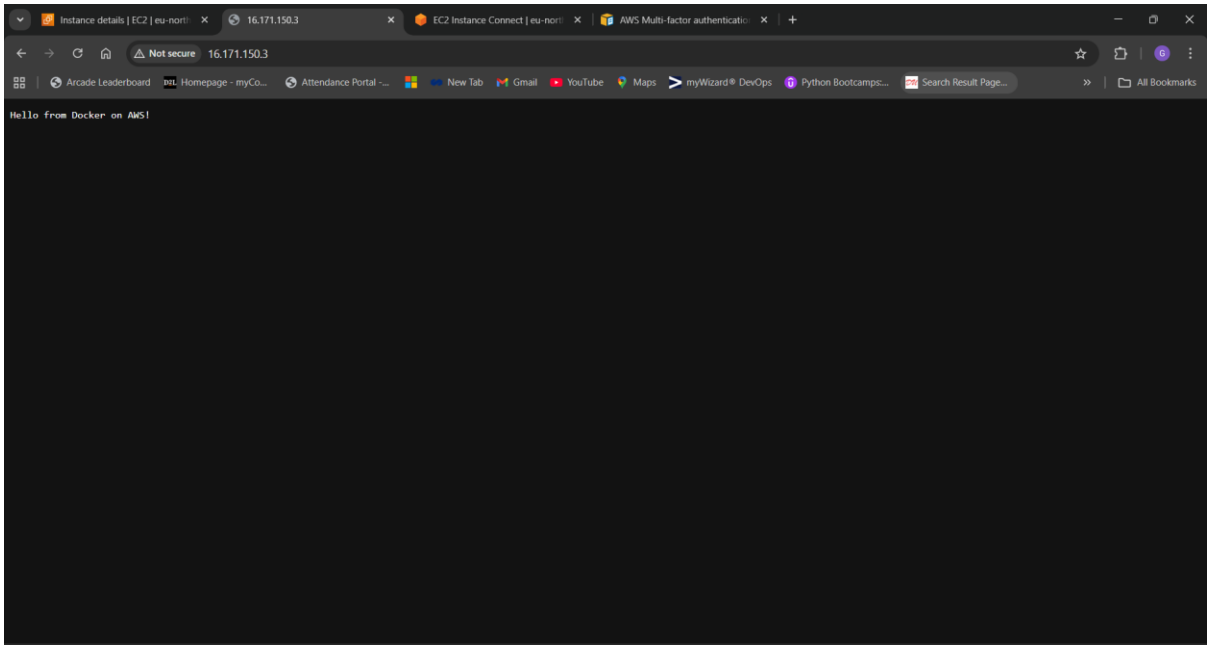
docker run -d -p 80:80 docker-sample-app

docker ps

```
50855e371badf9908b9d5155d6897b8b159983a38e260908d0e36635681ebec9ocker-sample-app
ubuntu@ip-172-31-38-27:~/docker-sample$ docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS          PORTS                                   NAMES
50855e371bad   docker-sample-app  "docker-entrypoint.s…"   20 seconds ago   Up 20 seconds   0.0.0.0:80->80/tcp, [::]:80->80/tcp   nostalgic_swanson
ubuntu@ip-172-31-38-27:~/docker-sample$
```

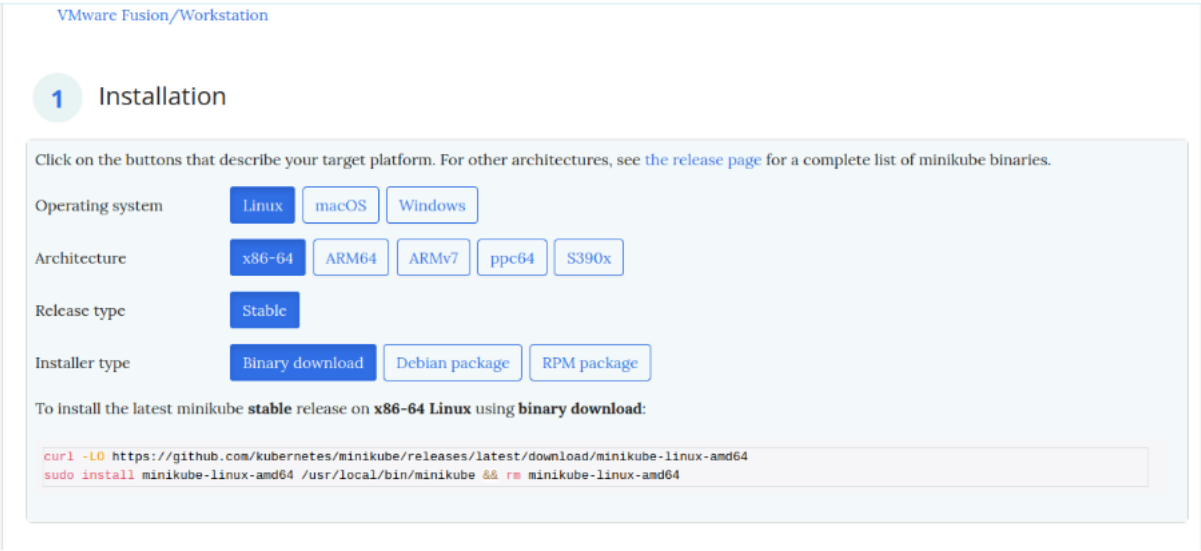
i-0eba8511aec755808 (docker-lab-instance)

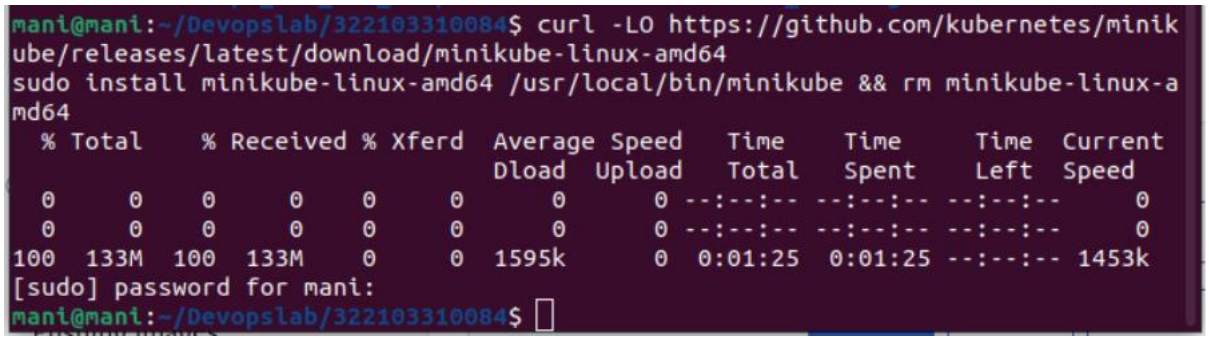
PublicIPs: 16.171.150.3 PrivateIPs: 172.31.38.27



b) Installing the Docker Engine with Minikube

Step 1: To install minikube, follow the instructions for your operating system at https://minikube.sigs.k8s.io/.





Step 2: To start minikube using the resource defaults and VirtualBox as the VM manager, enter the following in a terminal**.**

minikube start --driver=virtualbox

```
mani@mani:~/Devopslab/322103310084$ minikube start --driver=virtualbox
😄 minikube v1.37.0 on Ubuntu 22.04
✨ Using the virtualbox driver based on user configuration
💿 Downloading VM boot image ...
    > minikube-v1.37.0-amd64.iso....:  65 B / 65 B [---------] 100.00% ? p/s 0s
    > minikube-v1.37.0-amd64.iso:  370.78 MiB / 370.78 MiB  100.00% 5.79 MiB p/
👍 Starting "minikube" primary control-plane node in "minikube" cluster
💾 Downloading Kubernetes v1.34.0 preload ...
    > preloaded-images-k8s-v18-v1...:  337.07 MiB / 337.07 MiB  100.00% 6.21 Mi
🔥 Creating virtualbox VM (CPUs=2, Memory=3900MB, Disk=20000MB) ...
🐳 Preparing Kubernetes v1.34.0 on Docker 28.4.0 ...
🔗 Configuring bridge CNI (Container Networking Interface) ...
🔎 Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟 Enabled addons: default-storageclass, storage-provisioner
💡 kubectl not found. If you need it, try: 'minikube kubectl -- get pods -A'
🏄 Done! kubectl is now configured to use "minikube" cluster and "default" name
space by default
mani@mani:~/Devopslab/322103310084$
```

Installing the Docker Client and Setting Up Docker Environment Variables

Step 3: To install the docker client, follow the instructions at https://docs.docker.com/engine/install/binaries/ for your operating system.

```
mani@mani:~/Devopslab/322103310084$ tar /home/mani/Downloads/docker-28.5.1.tgz
tar: Old option 'g' requires an argument.
Try 'tar --help' or 'tar --usage' for more information.
mani@mani:~/Devopslab/322103310084$ tar xzvf /home/mani/Downloads/docker-28.5.1.
tgz
docker/
docker/runc
docker/docker
docker/dockerd
docker/containerd
docker/ctr
docker/containerd-shim-runc-v2
docker/docker-proxy
docker/docker-init
mani@mani:~/Devopslab/322103310084$ sudo cp docker/* /usr/bin/
[sudo] password for mani:
cp: cannot create regular file '/usr/bin/containerd': Text file busy
cp: cannot create regular file '/usr/bin/dockerd': Text file busy
```

```
mani@mani:~/Devopslab/322103310084$ sudo dockerd &
[1] 9773
mani@mani:~/Devopslab/322103310084$ INFO[2025-10-09T10:42:08.373378536+05:30] St
arting up
failed to start daemon, ensure docker is not running or delete /var/run/docker.p
id: process with PID 3446 is still running
^C
[1]+  Exit 1                  sudo dockerd
mani@mani:~/Devopslab/322103310084$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
17eec7bbc9d7: Pull complete
Digest: sha256:54e66cc1dd1fcb1c3c58bd8017914dbed8701e2d8c74d9262e26bd9cc1642d31
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

mani@mani:~/Devopslab/322103310084$
```

Step 4: In a terminal, enter the following to set your Docker environment variables.

eval $(minikube -p minikube docker-env)

```
mani@mani:~/Devopslab/322103310084$ eval $(minikube -p minikube docker-env)
mani@mani:~/Devopslab/322103310084$
```

Testing the Docker Client Connectivity

Step 5: Enter the following to check the Docker version.

docker version

```
mani@mani:~/Devopslab/322103310084$ docker version
Client:
 Version:           27.5.1
 API version:       1.47
 Go version:        go1.22.2
 Git commit:        27.5.1-0ubuntu3~22.04.2
 Built:             Mon Jun  2 12:18:38 2025
 OS/Arch:           linux/amd64
 Context:           default

Server:
 Engine:
  Version:          27.5.1
  API version:      1.47 (minimum version 1.24)
  Go version:       go1.22.2
  Git commit:       27.5.1-0ubuntu3~22.04.2
  Built:            Mon Jun  2 12:18:38 2025
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.7.27
  GitCommit:
 runc:
  Version:          1.2.5-0ubuntu1~22.04.1
  GitCommit:
 docker-init:
  Version:          0.19.0
  GitCommit:
```

Dissecting the Example telnet-server Dockerfile

Step 6: Navigate to the **telnet-server/** folder in the cloned repository and open the **Dockerfile**.

```
mani@mani: ~/Devopslab/322103310084/devops_for_the_des...
  GNU nano 6.2                    Dockerfile
# Build stage
FROM golang:alpine AS build-env
ADD . /
RUN cd / && go build -o telnet-server

# final stage
FROM alpine:latest as final
WORKDIR /app
ENV TELNET_PORT 2323
ENV METRIC_PORT 9000
COPY --from=build-env /telnet-server /app/

ENTRYPOINT ["./telnet-server"]
```

Step 7: The build stage uses the **golang:alpine** parent image named **build-env** to compile the Go-based ***telnet-server*** binary from source, and the final stage creates a minimal **Alpine** runtime image where the binary is copied into **/app**, environment variables set the server and

metrics ports (**2323** and **9000**), and the **ENTRYPOINT** runs the application, resulting in a lightweight, production-ready container.

```
mani@mani: ~/Devopslab/322103310084/devops_for_the_des...
  GNU nano 6.2                    Dockerfile
# Build stage
FROM golang:alpine AS build-env
ADD . /
RUN cd / && go build -o telnet-server

# final stage
FROM alpine:latest as final
WORKDIR /app
ENV TELNET_PORT 2323
ENV METRIC_PORT 9000
COPY --from=build-env /telnet-server /app/

ENTRYPOINT ["./telnet-server"]
```

Building the Container Image

Step 8: Enter the following to pass Docker the image name and Dockerfile location.

docker build -t dftd/telnet-server:v1 .

```
mani@mani:~/Devopslab/322103310084/devops_for_the_desperate/telnet-server$ docke
r build -t dftd/telnet-server:v1 .
DEPRECATED: The legacy builder is deprecated and will be removed in a future rel
ease.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  32.26kB
Step 1/9 : FROM golang:alpine AS build-env
alpine: Pulling from library/golang
2d35ebdb57d9: Pulling fs layer
c35fb4624d26: Pulling fs layer
4e2aec7ef170: Pulling fs layer
333b49ad6fbf: Pulling fs layer
4f4fb700ef54: Pulling fs layer
333b49ad6fbf: Waiting
4f4fb700ef54: Waiting
```

```
333b49ad6fbf: Waiting
4f4fb700ef54: Waiting
c35fb4624d26: Verifying Checksum
c35fb4624d26: Download complete
333b49ad6fbf: Verifying Checksum
333b49ad6fbf: Download complete
2d35ebdb57d9: Download complete
2d35ebdb57d9: Pull complete
c35fb4624d26: Pull complete
4f4fb700ef54: Download complete
4e2aec7ef170: Verifying Checksum
4e2aec7ef170: Download complete
4e2aec7ef170: Pull complete
333b49ad6fbf: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:182059d7dae0e1dfe222037d14b586ebece3ebf9a873a0fe1cc32e53dbea04e0
Status: Downloaded newer image for golang:alpine
 ---> f87d2639c296
Step 2/9 : ADD . /
 ---> ef08f7722052
Step 3/9 : RUN cd / && go build -o telnet-server
 ---> Running in 47f15d074f1e
go: downloading github.com/prometheus/client_golang v1.6.0
go: downloading github.com/beorn7/perks v1.0.1
go: downloading github.com/cespare/xxhash/v2 v2.1.1
go: downloading github.com/golang/protobuf v1.4.0
go: downloading github.com/prometheus/client_model v0.2.0
go: downloading github.com/prometheus/common v0.9.1
go: downloading github.com/prometheus/procfs v0.0.11
go: downloading google.golang.org/protobuf v1.21.0
go: downloading github.com/matttproud/golang_protobuf_extensions v1.0.1
go: downloading golang.org/x/sys v0.0.0-20200420163511-1957bb5e6d1f
 ---> Removed intermediate container 47f15d074f1e
 ---> 21c983b74e54
Step 4/9 : FROM alpine:latest as final
latest: Pulling from library/alpine
2d35ebdb57d9: Already exists
Digest: sha256:4b7ce07002c69e8f3d704a9c5d6fd3053be500b7f1c69fc0d80990c2ad8dd412
Status: Downloaded newer image for alpine:latest
 ---> 706db57fb206
Step 5/9 : WORKDIR /app
 ---> Running in 930172693d3e
 ---> Removed intermediate container 930172693d3e
 ---> 344ae4a1dc8a
Step 6/9 : ENV TELNET_PORT 2323
 ---> Running in 87dfa5ac831d
 ---> Removed intermediate container 87dfa5ac831d
 ---> 231468d17b07
```

```
Step 7/9 : ENV METRIC_PORT 9000
 ---> Running in 48ce04bc4d43
 ---> Removed intermediate container 48ce04bc4d43
 ---> d798f9a19f21
Step 8/9 : COPY --from=build-env /telnet-server /app/
 ---> 7130d5bfdb87
Step 9/9 : ENTRYPOINT ["./telnet-server"]
 ---> Running in ce87e8ab53e6
 ---> Removed intermediate container ce87e8ab53e6
 ---> 8dd656a29c92
Successfully built 8dd656a29c92
Successfully tagged dftd/telnet-server:v1
mani@mani:~/Devopslab/322103310084/devops_for_the_desperate/telnet-server$
```

Verifying the Docker Image

Step 9: Enter the following to list the Docker telnet-server image.

docker image ls dftd/telnet-server:v1

```
mani@mani:~/Devopslab/322103310084/devops_for_the_desperate/telnet-server$ docke
r image ls dftd/telnet-server:v1
REPOSITORY          TAG       IMAGE ID       CREATED          SIZE
dftd/telnet-server  v1        8dd656a29c92   3 minutes ago    22.4MB
mani@mani:~/Devopslab/322103310084/devops_for_the_desperate/telnet-server$
```

Running the Container

Step 10: The next step is to create and run the telnet-server container from the image you just built. Do this by entering the following.

docker run -p 2323:2323 -d --name telnet-server dftd/telnet-server:v1

```
mani@mani:~/Devopslab/322103310084/devops_for_the_desperate/telnet-server$ docke
r run -p 2323:2323 -d --name telnet-server dftd/telnet-server:v1
8ce62df2e687480123f7a8c006af9da67474b235c509fe036083db9a90e9b307
mani@mani:~/Devopslab/322103310084/devops_for_the_desperate/telnet-server$
```

The Docker run command uses the **-p** flag to map host port **2323** to the container's port **2323** for the telnet-server application, the **-d** flag to run the container in detached (background) mode, and the **--name** flag to assign the container the explicit name **telnet-server**, ensuring predictable access and avoiding Docker's default randomly generated names.

Step 11: Enter the following to verify that the container is actually running. The optional filter flag (-f) narrows the output to the containers you specify.

docker container ls -f name=telnet-server

```
mani@mani:~/Devopslab/322103310084/devops_for_the_desperate/telnet-server$ docke
r container ls -f name=telnet-server
CONTAINER ID   IMAGE                    COMMAND              CREATED          STATU
S          PORTS                    NAMES
8ce62df2e687   dftd/telnet-server:v1    "./telnet-server"    2 minutes ago    Up 2
minutes    0.0.0.0:2323->2323/tcp   telnet-server
mani@mani:~/Devopslab/322103310084/devops_for_the_desperate/telnet-server$
```

Step 12: Enter the following in your terminal to stop the container.

docker container stop telnet-server

```
mani@mani:~/Devopslab/322103310084/devops_for_the_desperate/telnet-server$ docke
r container stop telnet-server
telnet-server
mani@mani:~/Devopslab/322103310084/devops_for_the_desperate/telnet-server$
```

Other Docker Client Commands

exec: The **exec** command allows you to run a command inside a container or interact with a container, as if you were logged in to a terminal session.

docker exec telnet-server env

```
mani@mani:~/Devopslab/322103310084/devops_for_the_desperate/telnet-server$ docke
r exec telnet-server env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=8ce62df2e687
TELNET_PORT=2323
METRIC_PORT=9000
HOME=/root
mani@mani:~/Devopslab/322103310084/devops_for_the_desperate/telnet-server$
```

docker exec -it telnet-server /bin/sh

```
mani@mani:~/Devopslab/322103310084/devops_for_the_desperate/telnet-server$ docke
r exec -it telnet-server /bin/sh
/app # ls
telnet-server
/app #
```

inspect: The **inspect** docker command returns low-level information about some Docker objects. The output is in **JSON** format by default.

docker inspect telnet-server

```
mani@mani:~/Devopslab/322103310084/devops_for_the_desperate/telnet-server$ docke
r inspect telnet-server
[
    {
        "Id": "8ce62df2e687480123f7a8c006af9da67474b235c509fe036083db9a90e9b307"
,
        "Created": "2025-10-09T05:56:27.855388349Z",
        "Path": "./telnet-server",
        "Args": [],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 16925,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2025-10-09T06:04:48.356131775Z",
            "FinishedAt": "2025-10-09T06:01:28.510225603Z"
```

stats: The **stats** command displays a real-time update on the resources a container is using.

docker stats --no-stream telnet-server

```
mani@mani:~/Devopslab/322103310084/devops_for_the_desperate/telnet-server$ docke
r stats --no-stream telnet-server
CONTAINER ID   NAME            CPU %     MEM USAGE / LIMIT     MEM %     NET I/O
       BLOCK I/O   PIDS
8ce62df2e687   telnet-server   0.00%     1.625MiB / 3.634GiB   0.04%     840B /
126B   0B / 0B     5
mani@mani:~/Devopslab/322103310084/devops_for_the_desperate/telnet-server$
```

history: The **history** command displays a container image's history, which is useful for viewing the number and sizes of an image's layers.

docker history dftd/telnet-server:v1

```
mani@mani:~/Devopslab/322103310084/devops_for_the_desperate/telnet-server$ docke
r history dftd/telnet-server:v1
IMAGE          CREATED          CREATED BY
SIZE      COMMENT
8dd656a29c92   20 minutes ago   /bin/sh -c #(nop)  ENTRYPOINT ["./telnet-ser…
0B
7130d5bfdb87   20 minutes ago   /bin/sh -c #(nop) COPY file:985f8f86fb7bf2a8…
14.1MB
d798f9a19f21   20 minutes ago   /bin/sh -c #(nop)  ENV METRIC_PORT=9000
0B
231468d17b07   20 minutes ago   /bin/sh -c #(nop)  ENV TELNET_PORT=2323
0B
344ae4a1dc8a   20 minutes ago   /bin/sh -c #(nop) WORKDIR /app
0B
706db57fb206   19 hours ago     CMD ["/bin/sh"]
0B        buildkit.dockerfile.v0
<missing>      19 hours ago     ADD alpine-minirootfs-3.22.2-x86_64.tar.gz /…
8.32MB    buildkit.dockerfile.v0
mani@mani:~/Devopslab/322103310084/devops_for_the_desperate/telnet-server$
```

rm: The **rm** command removes a stopped container.

docker container rm telnet-server

```
mani@mani:~/Devopslab/322103310084/devops_for_the_desperate/telnet-server$ docke
r container rm telnet-server
telnet-server
mani@mani:~/Devopslab/322103310084/devops_for_the_desperate/telnet-server$
```

Connecting to the Telnet-Server

Step 13: Enter the following in a terminal to get the IP address.

minikube ip

```
mani@mani:~/Devopslab/322103310084/devops_for_the_desperate/telnet-server$ minik
ube ip
192.168.59.100
mani@mani:~/Devopslab/322103310084/devops_for_the_desperate/telnet-server$
```

Step 14: To connect to the telnet-server running inside the container, pass the IP address (**192.168.59.100**) and port (**2323**) to the telnet command.

telnet 192.168.59.100 2323

```
mani@mani:~/Devopslab/322103310084/devops_for_the_desperate/telnet-server$ telne
t 192.168.59.100 2323
Trying 192.168.59.100...
Connected to 192.168.59.100.
Escape character is '^]'.

>
```

Step 15: While still connected to the telnet-server, enter the following to print the current date and time.

d

```
>d
Fri Oct 10 01:56:57 +0000 UTC 2025
```

Step 16: To connect to the telnet-server running inside the container, pass the IP address (**192.168.59.100**) and port (**2323**) to the telnet command.

q

```
>q
Good Bye!
Connection closed by foreign host.
mani@mani:~/Devopslab/322103310084/devops_for_the_desperate/telnet-server$
```

Connecting to the Telnet-Server

Step 17: To see all the logs for the telnet-server, which is logging to STDOUT, enter the following in your terminal.

docker logs telnet-server

```
mani@mani:~/Devopslab/322103310084/devops_for_the_desperate/telnet-server$ docke
r logs telnet-server
telnet-server: 2025/10/10 01:52:38 Metrics endpoint listening on :9000
telnet-server: 2025/10/10 01:52:38 telnet-server listening on [::]:2323
telnet-server: 2025/10/10 01:55:13 [IP=192.168.59.1] New session
telnet-server: 2025/10/10 01:56:57 [IP=192.168.59.1] Requested command: d
telnet-server: 2025/10/10 01:56:59 [IP=192.168.59.1] User quit session
mani@mani:~/Devopslab/322103310084/devops_for_the_desperate/telnet-server$
```